UNITED STATES PATENT APPLICATION

FOR

FIXED TIMEFRAME CACHE TAG ARRAY ARBITRATION MECHANISM FOR INVALIDATIONS

INVENTOR:

GEORGE Z. CHRYSOS

DOCKET NO. 42P17892

PREPARED BY:

AMI PATEL SHAH

REG. NO. 42,143

FIXED TIMEFRAME CACHE TAG ARRAY

ARBITRATION MECHANISM FOR INVALIDATIONS

BACKGROUND INFORMATION

[0001] To maintain memory coherence, a cache must process invalidation requests to blocks that are shared (S in MESI). This requires access to a cache tag array to determine whether the targeted block is in the cache. However, other operations that require tag array access can also be occurring simultaneously (fills, reads, or writes). Since providing dedicated ports into the tag array for all possible users of the tag array in a given cycle is expensive (generally area increases as the number of ports squared, and power consumption increases linearly with area), arbitration amongst multiple user for fewer ports is necessary.

BRIEF DESCRIPTION OF THE DRAWINGS

[0002] Various features of the invention will be apparent from the following description of preferred embodiments as illustrated in the accompanying drawings, in which like reference numerals generally refer to the same parts throughout the drawings. The drawings are not necessarily to scale, the emphasis instead being placed upon illustrating the principles of the inventions.

[0003] Figure 1 is a flowchart for a fixed timeframe cache tag array arbitration mechanism for invalidations.

DETAILED DESCRIPTION

[0004] In the following description, for purposes of explanation and not limitation, specific details are set forth such as particular structures, architectures, interfaces, techniques, etc. in order to provide a thorough understanding of the various aspects of the invention. However, it will be apparent to those skilled in the art having the benefit of the present disclosure that the various aspects of the invention may be practiced in other examples that depart from these specific details. In certain instances, descriptions of well-known devices, circuits, and methods are omitted so as not to obscure the description of the present invention with unnecessary detail.

[0005] Each processor core may have its own private cache. The cache has a tag array that contains information about which blocks are in the cache. Each invalidate request has an address of the block associated with it and the address consists of an index to the tag array. The tag array may be accessed for multiple reasons. One reason may be during a invalidate request, the processor searches its cache to determine if it has the requested block. If the cache has the block, it has to invalidate the requested block. Now new loads coming into an execution pipeline will not be able to hit the cache.

[0006] A second reason may be during a fill request where the tag array is updated. When a fill request is sent, a block is requested, the block is obtained, the block is filled into the cache and then the tag is written to. A fill and invalidate request may happen simultaneously. They can also happen every cycle. The following mechanism focuses on conflicts between fills and invalidations happening simultaneously, but, the mechanism described can be generally applied to invalidations conflicting with any other operation type.

[0007] Figure 1 is a flowchart for a fixed timeframe cache tag array arbitration mechanism for invalidations. Initially, a invalidate request is sent into a pipeline. This means that a targeted cache is being sent the address of the block in question. In step 100, an orchestrating agent determines if any fills are also coming into the pipeline. If no fills are coming into the pipeline, then the targeted cache performs a tag array lookup and compares the addresses found in the indexed set to the sent address. If the tags match, the targeted cache clears the valid bit in the tag array to prevent subsequent hits on that block in the cache (step 110). If there is a fill, then there is a collision of a fill and invalidate occurring in the same cycle. In this case, invalidate request is pipelined to the first cycle (step 120). The orchestrating agent again determines if there is a fill in step 130. If no fill is occurring, then step 110 is repeated, otherwise, the invalidate request is pipelined to the second cycle (step 140). Again it is determined if a fill has occurred in step 150. If not, step 110 is repeated. Otherwise, the whole index set is now invalidated in step 160. The index bits of the invalidated block can simply clear the valid bits of every block in the indexed set (guaranteeing that if the block was resident in the cache, it is now invalid).

[0008] This above described mechanism gives invalidations several “tries” to read the tag array and perform a precise invalidation (the tag matches the invalidation block address). If in a given cycle, an invalidation request conflicts with a fill request or even an older invalidation, the invalidation request is piped and tried again in the next cycle. After some number of tries, the invalidation will give up on trying to read the tag array and just perform a invalidation on every block in the set. The number of tries can be arbitrarily lengthened to provide less probability of the full set invalidation, with the provision that the time that the core guarantees invalidation of a block will then extend, and may delay granting ownership of the block to another CPU.

[0009] By allowing three tries to invalidate, the present mechanism builds more tolerance into a cache coherent computer system. It allows the invalidation three chances to actually do a real lookup into a port and guarantee that invalidation has been completed. After those three chances expire, the whole index set is invalidated.

[0010] In the following description, for purposes of explanation and not limitation, specific details are set forth such as particular structures, architectures, interfaces, techniques, etc. in order to provide a thorough understanding of the various aspects of the invention. However, it will be apparent to those skilled in the art having the benefit of the present disclosure that the various aspects of the invention may be practiced in other examples that depart from these specific details. In certain instances, descriptions of well-known devices, circuits, and methods are omitted so as not to obscure the description of the present invention with unnecessary detail.